

A Geometric Approach to the Problem of Unique Decomposition of Processes

Thibaut Balabonski¹ and Emmanuel Haucourt²

¹ Laboratoire PPS, Université Paris Diderot and CNRS, UMR 7126
thibaut.balabonski@pps.jussieu.fr

² CEA, LIST, Gif-sur-Yvette, F-91191, France.
emmanuel.haucourt@cea.fr

Abstract. This paper proposes a geometric solution to the problem of prime decomposability of concurrent processes first explored by R. Milner and F. Moller in [MM93]. Concurrent programs are given a geometric semantics using cubical areas, for which a unique factorization theorem is proved. An effective factorization method which is correct and complete with respect to the geometric semantics is derived from the factorization theorem. This algorithm is implemented in the static analyzer **ALCOOL**.

1 Introduction: Parallel Programming Problem

This paper aims at introducing some new static analysis technology for concurrent programs. The work presented here gives a new insight into the problem of decomposition of processes, which was first explored by *R. Milner* and *F. Moller* in [MM93]. The main new results are an algorithm maximally decomposing concurrent programs into independent processes (Section 4) and the proof that this prime decomposition is unique in the considered class of programs (Theorem 2). They are derived from a study of algebraic properties of cubical areas.

Given an associative and commutative operator \parallel for parallel composition of two processes (with the empty process as unit), decomposing a concurrent program P into a multiset $\{P_1, \dots, P_n\}$ such that $P = P_1 \parallel \dots \parallel P_n$ and the P_i s are independent has several interests. For instance the decomposition may be relevant for the allocation of processors to subprograms. Another important concern is the static analysis of concurrent programs, whose complexity grows exponentially with the number of concurrent processes: finding independent subprograms that can be analyzed separately could dramatically decrease the global complexity of the static analysis. Hence this paper aims at finding the finest decomposition (and proving its existence) for a wide class of concurrent programs.

Let us first take a look at a non trivial example of independent processes, in the so-called *PV* language introduced by *E.W. Dijkstra* [Dij68] as a simple

² This work has been partially supported by *Agence Nationale pour la Recherche* via the project PANDA (Parallel and Distributed Analysis) ANR-09-BLAN-0169-02

framework for the study of concurrency with shared resources. The only instructions are $P(name)$ and $V(name)$ ³, where $name$ is an identifier which refers to a resource. The idea is to have some common pool of resources which can be taken (with P) and released (with V) by concurrent processes. The resources are formalized by semaphores which, depending on their arity, can be held simultaneously by a certain number of processes (arity n allows at most $n - 1$ simultaneous processes).

Now suppose c is the name of a ternary semaphore, which means it can be held by at most two processes, and a, b are the names of binary semaphores, also called *mutex* for *mutual exclusion*.

Example 1.

$$\begin{aligned} \Sigma &:= \pi_1 = Pa.Pc.Vc.Va \\ &\parallel \pi_2 = Pb.Pc.Vc.Vb \\ &\parallel \pi_3 = Pa.Pc.Vc.Va \\ &\parallel \pi_4 = Pb.Pc.Vc.Vb \end{aligned}$$

A naive syntactic analysis would stamp this program as undecomposable since all processes share the resource c , but the following finer analysis can be made: thanks to mutex a (respectively b), the processes π_1 and π_3 (respectively π_2 and π_4) cannot both hold an occurrence of the resource c at the same time. Then there are never more than two simultaneous requests over c , which means that the instructions Pc and Vc play actually no role in determining the semantics of the program. And without c , Σ can be split in two independent systems (they use disjoint resources). Basically, this example is based on the fact that semaphores are not the real resources, but mere devices used to guard their access. And it may be that some guards are redundant.

This work is based on a geometric semantics for concurrency. The semantics for PV programs was implicitly given in [Dij68], then explicited by Carson et al.[CR87]. Roughly speaking, the instructions of a process are pinned upon a 1-dimensional “directed” shape, in other words track along which the instructions of the program to execute are written. If N sequential processes run together, one can consider their N instruction pointers as a multi-dimensional control point.

Although we have made the construction explicit for PV programs only, the result applies to any synchronisation or communication mechanism whose geometric interpretation is a so-called *cubical area* (the notion is formalized in Section 3.5). See for instance [GH05] for the geometric semantics of synchronisation barriers, monitors and synchronous or asynchronous communications (with finite or infinite message queues): their geometrical shape is the complement of an orthogonal polyhedron [BMP99,Tha09], which is a special case of cubical area.

Outline of the paper.

The paper is organized as follows. Section 2 provides the mathematics of the geometric semantics, detailed for PV programs. Section 3 establishes the link

³ P and V stand for the dutch words “Pakken” (take) and “Vrijlaten” (release)

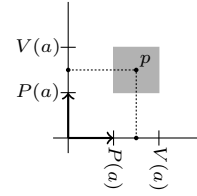
between algebraic properties of the semantics and independence of subprograms, and then states and proves prime decomposability theorems for algebraic frameworks encompassing the geometric semantics (Theorems 1 and 2). Section 4 describes the corresponding algorithm and implementation as well as a detailed example and some benchmarks.

2 The Geometric Semantics

The geometric semantics of a PV program is a subset of the finite dimensional real vector space whose dimension is roughly speaking the number N of processes running concurrently. Then each process is associated with a coordinate of \mathbb{R}^N . Yet given a mutex \mathbf{a} , the instructions $P(\mathbf{a})$ and $V(\mathbf{a})$ that occur in the k^{th} process should be understood as opening and closing parentheses or more geometrically as the least upper bound and the greatest lower bound of an interval I_k of \mathbb{R} . The forbidden area generated by a mutex \mathbf{a} is thus the finite union of hyperrectangles⁴ of the following form (with $k < k'$)

$$\underbrace{\mathbb{R}^+ \times \cdots \times \mathbb{R}^+ \times I_k \times \mathbb{R}^+ \times \cdots \times \mathbb{R}^+ \times I_{k'} \times \mathbb{R}^+ \times \cdots \times \mathbb{R}^+}_{\text{product of } N \text{ terms}}$$

For example, $P(\mathbf{a}) \cdot V(\mathbf{a}) \parallel P(\mathbf{a}) \cdot V(\mathbf{a})$ is a program written in PV language. Assuming that \mathbf{a} is a mutex (semaphore of arity 2), its geometric model is $(\mathbb{R}^+)^2 \setminus [1, 2]^2$. Intuitively, a point p in $[1, 2]^2$ would correspond to the situation where both processes hold the semaphore \mathbf{a} , which is forbidden by the semantics of mutexes.



In the sequel of this section we formalize the PV language syntax as well as the construction of the geometric semantics. Denote the positive half-line $[0, +\infty[$ by \mathbb{R}^+ . For each $\alpha \in \mathbb{N} \setminus \{0, 1\}$ let S_α be an infinite countable set whose elements are the semaphores of arity α of the PV language. A **PV process** is a finite sequence on the alphabet

$$A := \{P(s), V(s) \mid s \in \bigcup_{\alpha \geq 2} S_\alpha\}$$

and a **PV program** is a finite (and possibly empty) multiset of PV processes. The parallel operator then corresponds to the multiset addition therefore it is associative and commutative⁵. Given a semaphore s and a process π , the sequences $(x_k)_{k \in \mathbb{N}}$ and $(y_k)_{k \in \mathbb{N}}$ are recursively defined as follows: set $y_{-1} = 0$ and

⁴ however we will more likely write “cube” instead.

⁵ The collection of multisets over a set \mathbb{A} forms a monoid which is isomorphic to the free commutative monoid over \mathbb{A} . The first terminology is usually used by computer scientists while mathematicians prefer the second one. Anyway it will be described and characterized in the Section 3.

- $x_k = \min\{n \in \mathbb{N} \mid n > y_{k-1} \text{ and } \pi(n) \text{ is } P(s)\}$
- $y_k = \min\{n \in \mathbb{N} \mid n > x_k \text{ and } \pi(n) \text{ is } V(s)\}$

with the convention that $\min \emptyset = \infty$, $\pi(n)$ denotes the n^{th} term of the process π and its first term is $\pi(1)$. Then, the **busy area** of s in π is⁶

$$B_s(\pi) := \bigcup_{k \in \mathbb{N}} [x_k, y_k[$$

Actually this description requires some extra assumptions upon the way instructions are interpreted. Namely a process cannot hold more than one occurrence of a given resource. Thus a process already holding an occurrence of a semaphore s ignores any instruction $P(s)$, and similarly a process holding no occurrence of s ignores any instruction $V(s)$. Then denote by $\chi_s^\pi : \mathbb{R} \rightarrow \mathbb{R}$ the characteristic function of B_s defined by

$$\chi_s^\pi(x) = \begin{cases} 1 & \text{if } x \in B_s(\pi) \\ 0 & \text{otherwise} \end{cases}$$

Because the sequence π is finite, there exists some k such that $x_k = \infty$ and for any such k and any $k' \geq k$, one also has $x_{k'} = \infty$. In particular, if the instruction $P(s)$ does not appear in π , then $B_s(\pi)$ is empty and χ_s^π is the null map. The geometric model of a PV program with N processes running concurrently is a subpospace of $[0, +\infty[^N$ defined as follows:

- Call $\Pi = (\pi_1, \dots, \pi_N)$ the program to modelize.
- Given a semaphore s of arity α define the **forbidden area** of s in Π as

$$F_s := \{ \vec{x} \in [0, +\infty[^N \mid \vec{\chi}_s \cdot \vec{x} \geq \alpha \}$$

where $\vec{x} = (x_1, \dots, x_N)$, $\vec{\chi}_s = (\chi_s^{\pi_1}, \dots, \chi_s^{\pi_N})$ and $\vec{\chi}_s \cdot \vec{x} = \sum_{i=1}^N \chi_s^{\pi_i}(x_i)$. The value $\vec{\chi}_s \cdot \vec{x}$ indicates how many occurrences of the semaphore s are held when the instruction pointer is at position \vec{x} . Note that F_s is a finite union of hyperrectangles which may be empty even if s appears in the program Π . In the end, the **forbidden area** of the program Π is the following union over S the union of all the sets S_α .

$$F := \bigcup_{s \in S} F_s$$

Because there are finitely many resource names s appearing in a PV program, there are finitely many non empty set F_s . Hence the previous union is still a finite union of hyperrectangles. The **state space** or **geometric model** of Π is then $[0, +\infty[^N \setminus F$, and is denoted by $\llbracket \Pi \rrbracket$. Remark that the geometric model is also a finite union of hyperrectangles.

⁶ Including the greatest lower bound and removing the least upper one is the mathematical interpretation of the following convention: the changes induced by an instruction are effective exactly when the instruction pointer reaches it.

In other words, the state space of Π is the set of positions of the “multi dimensional instruction pointer” for which the number of occurrences of each semaphore s is strictly below its arity α . If Π is made of N concurrent process, this space is a N -dimensional euclidean space with (cubical) holes. As an example, Figure 1 shows the construction of the geometric model of the PV program $P(a)P(b)V(b)V(a) \parallel P(b)P(a)V(a)V(b)$ (referred to as the *swiss flag*). Figure 2 gives a simplified version of Example 1 fitting in three dimensions.

Fig. 1. Construction of a geometric model: the swiss flag

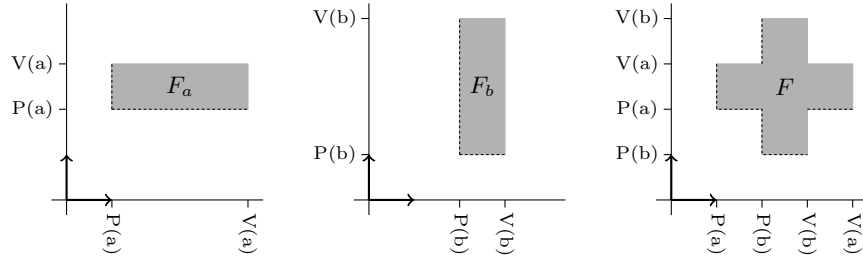
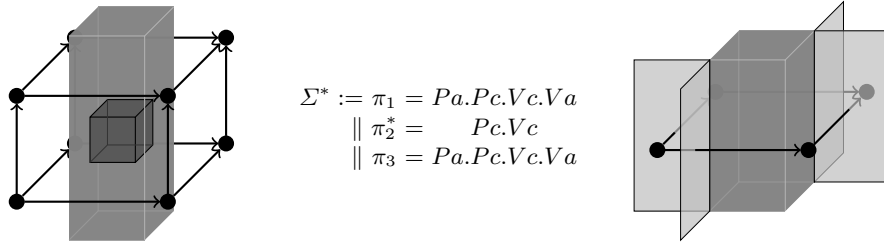


Fig. 2. Example in three dimensions



Intuitively, the graphs pictured here correspond to the *essential components* of the state space, see [GH07] for developments on this topic. The dark grey cube on the left picture is the forbidden area of the semaphore c , which is contained in the forbidden area of the mutex a (in the full –and 4D– example Σ the forbidden area of c is contained in the union of the forbidden areas of a and b).

3 The Problem of Unique Decomposition

Now that the geometric semantics of programs is defined, let us refocus on the main goal: finding the independent parts of a concurrent program. Hence the question: what does independence mean in this geometrical setting?

3.1 Parallel Composition vs Cartesian Product

A general definition has to be given for **independence**: say a program Π is independent from another program Π' when its behaviour is unaffected by parallel composition with Π' , whatever the way Π' is executed. That means, the presence of Π' , as well as its instruction pointer, has no effect on the semantics of Π . A geometric translation of this assertion is: in the geometric model of $\Pi \parallel \Pi'$, the cylinder⁷ over any state of Π' (i.e. the subspace of all points with given fix coordinates for the Π' component) is equal to the model of Π .

Hence two programs Π and Π' of geometric models $\llbracket \Pi \rrbracket$ and $\llbracket \Pi' \rrbracket$ are independent if and only if the geometric model $\llbracket \Pi \parallel \Pi' \rrbracket$ of their parallel composition is isomorphic to the cartesian product $\llbracket \Pi \rrbracket \times \llbracket \Pi' \rrbracket$. Thus the decompositions of a program correspond to the factorizations of its geometric model (with respect to the cartesian product). Next subsection reminds some algebraic settings and results needed for a notion like *factorization* to make sense.

3.2 Free Commutative Monoids

The reader not familiar with this notion can refer for instance to [Lan02]. Let M be a commutative monoid. Any element of M which has an inverse is called a **unit**. A *non unit* element x of M is said to be **irreducible** when for all y and z in M , if $x = yz$ then y or z is a unit. The set of irreducible elements of M is denoted by $I(M)$.

For any elements x and y of M , say x **divides** y when there is an element x' of M such that $xx' = y$. A *non unit* element x of M is said to be **prime** when for all y and z in M , if x divides yz then x divides y or x divides z . The set of prime elements of M is denoted by $P(M)$.

Given a set X , the collection of maps ϕ from X to \mathbb{N} such that $\{x \in X \mid \phi(x) \neq 0\}$ is finite, together with the pointwise addition, forms a commutative monoid whose neutral element is the null map: we denote it by $F(X)$. Yet, given any subset X of a commutative monoid M , the following map

$$\begin{array}{ccc} \Phi_M^X : F(X) & \longrightarrow & M \\ \phi & \longmapsto & \prod_{x \in X} x^{\phi(x)} \end{array}$$

is a well-defined morphism of monoids. A well-known result asserts that the following are equivalent [Lan02]:

⁷ Categorists would write “fibre” instead of “cylinder”.

1. the mapping $\Phi_M^{I(M)}$ is an isomorphism of monoids
2. the set $I(M)$ generates⁸ M and $I(M) = P(M)$
3. any element of M can be written as a product of irreducible elements in a unique way up to permutation of terms (unique decomposition property).

In this case M is said to be a **free** commutative monoid.

Two standard examples of free commutative monoids are given by the set of non zero natural numbers $\mathbb{N} \setminus \{0\}$ together with multiplication (the unit is 1 and the irreducible elements are the prime numbers) and the set of natural numbers \mathbb{N} together with addition (the unit is 0 and the only irreducible element is 1).

However, neither the multiplicative monoid $\mathbb{Z} \setminus \{0\}$ nor the additive group \mathbb{Z} are free commutative monoids since they both contain a non trivial unit, namely -1 in both cases.

Also note that all the non zero elements of the additive monoid \mathbb{R}_+ are primes though it does not contain a single irreducible element.

A more intricate phenomenon arises with polynomials [HN50]: the (multiplicative) commutative monoid $\mathbb{N}[X] \setminus \{0\}$ of non zero polynomials with natural coefficients is not free. Indeed, although each element of $\mathbb{N}[X] \setminus \{0\}$ is a product of irreducible polynomials these decompositions are not unique: we have

$$(1 + X)(1 + X^2 + X^4) = (1 + X^3)(1 + X + X^2)$$

where all of the polynomials $1 + X$, $1 + X^2 + X^4$, $1 + X^3$ and $1 + X + X^2$ are irreducible (which is not the case in the monoid of polynomials with coefficients in \mathbb{Z} , indeed the ring $\mathbb{Z}[X]$ is **factorial** [Lan02]).

3.3 Cartesian Product and Commutation

The geometric model of a concurrent program is a set of points in an euclidean space of finite dimension. Thus each point can be represented by the tuple of its coordinates, and a geometric model becomes a set of tuples (of same length which corresponds to the dimension of the space). The cartesian product on such structures is the following:

$$X \times Y = \{ (x_1, \dots, x_n, y_1, \dots, y_k) \mid (x_1, \dots, x_n) \in X, (y_1, \dots, y_k) \in Y \}$$

However, this operator is not commutative whereas the parallel composition of programs should be so. Thus, in order to model parallel composition, we make the operator \times commutative monoid through quotient by permutation of coordinates. In the next subsection we prove a freeness theorem for a monoid generalizing this idea: tuples of (real) coordinates are replaced by words over an arbitrary (potentially infinite) alphabet. The geometric model of a PV program therefore belongs to a free commutative monoid and thus admits a unique decomposition of irreducible elements, from which the processes factorization is deduced.

⁸ $X \subseteq M$ generates M when all its elements can be written as a product of elements of X . The product of the empty set being defined as the neutral element. Remark then that “ $I(M)$ generates M ” implies that the only unit of M is its neutral element.

3.4 Homogeneous Sets of Words

Let \mathbb{A} be a set called the alphabet. The *non commutative* monoid of words \mathbb{A}^* consists on the finite sequences of elements of \mathbb{A} together with **concatenation**. Given words w and w' of length n and n' , the word $w * w'$ of length $n + n'$ is defined by

$$(w * w')_k = \begin{cases} w_k & \text{if } 1 \leq k \leq n \\ w'_{k-n} & \text{if } n+1 \leq k \leq n+n' \end{cases}$$

The length of a word w is also referred to as $\ell(w)$. A **subword** of w is a word of the form $w \circ \phi$ where ϕ is a strictly increasing map $\{1, \dots, n\} \rightarrow \{1, \dots, \ell(w)\}$. Hence a subword of w is also entirely characterized by the image of the increasing map ϕ i.e. by a subset of $\{1, \dots, \ell(w)\}$. If A is the image of ϕ then we write $w \circ A$ instead of $w \circ \phi$.

The n^{th} *symmetric group* \mathfrak{S}_n (the group of permutations of the set $\{1, \dots, n\}$) acts on the set of words of length n by composing on the right, that is for all $\sigma \in \mathfrak{S}_n$ and all word w of length n we have

$$\sigma \cdot w := w \circ \sigma = (w_{\sigma(1)} \cdots w_{\sigma(n)})$$

The concatenation extends to sets of words. Given $S, S' \subseteq \mathbb{A}^*$, define

$$S * S' := \{w * w' \mid w \in S; w' \in S'\}$$

Remark that this concatenation of sets corresponds to the cartesian product.

The set $\mathcal{P}(\mathbb{A}^*)$ of subsets of \mathbb{A}^* is thus endowed with a structure of *non commutative* monoid whose neutral element is $\{\epsilon\}$: the singleton containing the empty word. Note that the empty set \emptyset is the *absorbing* element of $\mathcal{P}(\mathbb{A}^*)$, that is for all $S \subseteq \mathbb{A}^*$ we have

$$\emptyset * S = S * \emptyset = \emptyset$$

A subset H of \mathbb{A}^* is said to be **homogeneous** when all the words it contains share the same length n . By analogy with the geometric construction, n is called the dimension of H and denoted by $d(H)$. The symmetric group \mathfrak{S}_n acts on the set of homogeneous set of dimension n in a natural way by applying the same permutation to all words:

$$\sigma \cdot H := \{\sigma \cdot w \mid w \in H\}$$

The homogeneous subsets of \mathbb{A}^* form a sub-monoid $\mathcal{P}_h(\mathbb{A}^*)$ of $\mathcal{P}(\mathbb{A}^*)$ and can be equipped with an equivalence relation as follows: write $H \sim H'$ when $d(H) = d(H') = n$ and there exists $\sigma \in \mathfrak{S}_n$ such that $H' = \sigma \cdot H$. Moreover, for two permutations $\sigma \in \mathfrak{S}_n$ and $\sigma' \in \mathfrak{S}_{n'}$, define the **juxtaposition** $\sigma \otimes \sigma' \in \mathfrak{S}_{n+n'}$ as:

$$\sigma \otimes \sigma'(k) := \begin{cases} \sigma(k) & \text{if } 1 \leq k \leq n \\ (\sigma'(k-n)) + n & \text{if } n+1 \leq k \leq n+n' \end{cases}$$

A Godement-like exchange law is satisfied, which ensures that \sim is actually a congruence:

$$(\sigma \cdot H) * (\sigma' \cdot H') = (\sigma \otimes \sigma') \cdot (H * H')$$

Hence the quotient $\mathcal{P}_h(\mathbb{A}^*)/\sim$ from which the absorbing element has been removed is still a monoid called the **homogeneous monoid** over \mathbb{A} and denoted by $\mathcal{H}(\mathbb{A})$. Moreover the homogeneous monoid is commutative and its only unit is the singleton $\{\epsilon\}$. Remark that if the alphabet \mathbb{A} is a singleton (resp. the empty set) then the homogeneous monoid $\mathcal{H}(\mathbb{A})$ is isomorphic to $(\mathbb{N}, +, 0)$ (resp. the null monoid).

Theorem 1. *For any set \mathbb{A} the homogeneous monoid over \mathbb{A} is free.*

Proof. We check the conditions 1-3 which characterize the free commutative monoids (see Section 3.2). Since $d(H * H') = d(H) + d(H')$ we deduce from a straightforward induction on the dimension of elements of $\mathcal{H}(\mathbb{A})$ that they can all be written as products of irreducible elements: $I(\mathcal{H}(\mathbb{A}))$ generates $\mathcal{H}(\mathbb{A})$.

Now suppose H is an irreducible element of $\mathcal{H}(\mathbb{A})$ which divides $H_1 * H_2$ and pick S, S_1 and S_2 respectively from the equivalence classes H, H_1 and H_2 . Define $n = d(H)$, $n_1 = d(H_1)$ and $n_2 = d(H_2)$, and remark that $n = n_1 + n_2$. There exists $\sigma \in \mathfrak{S}_n$ and some S_3 such that $\sigma \cdot (S_1 * S_2) = S * S_3$ in $\mathcal{P}_h(\mathbb{A}^*)$. Suppose in addition that H does not divide H_1 nor H_2 , then we have $A_1 \subseteq \{1, \dots, n_1\}$ and $A_2 \subseteq \{1, \dots, n_2\}$ s.t. $A_1 \neq \emptyset$, $A_2 \neq \emptyset$ and $\sigma(A_1 \cup A_2) = \{1, \dots, n\}$ where $A'_2 := \{a + n_1 \mid a \in A_2\}$. Then we have a non trivial factoring $S = S'_1 * S'_2$ where

$$S'_1 := \{w \circ A_1 \mid w \in S_1\} \text{ and } S'_2 := \{w \circ A_2 \mid w \in S_2\}$$

This contradicts irreducibility of H . Hence H divides H_1 or H_2 and thus H is prime. So any irreducible element of $\mathcal{H}(\mathbb{A})$ is prime: $I(\mathcal{H}(\mathbb{A})) \subseteq P(\mathcal{H}(\mathbb{A}))$.

Finally, suppose H is a prime element of $\mathcal{H}(\mathbb{A})$ such that $H = H_1 * H_2$. In particular H divides $H_1 * H_2$, and since H is prime it divides H_1 or H_2 . Both cases being symmetrical, suppose H divides H_1 . In particular $d(H) \leq d(H_1)$. On the other hand $d(H) = d(H_1) + d(H_2)$, and thus $d(H_2) \leq 0$. Dimensions being natural numbers, we deduce that $d(H_2) = 0$ and then that $H_2 = \{\epsilon\}$. Hence H is irreducible, and $I(\mathcal{H}(\mathbb{A})) = P(\mathcal{H}(\mathbb{A}))$.

One of the worthy feature of the construction is that any binary relation \diamond over $\mathcal{P}_h(\mathbb{A})$ which is compatible with the product and satisfies

$$\forall S, S' \in \mathcal{P}_h(\mathbb{A}) \quad (d(S) = d(S') = n \text{ and } S \diamond S' \Rightarrow \forall \sigma \in \mathfrak{S}_n \quad (\sigma \cdot S) \diamond (\sigma \cdot S'))$$

can be extended to a relation on $\mathcal{H}(\mathbb{A})$ which is still compatible with the product. Actually it suffices to set $H \diamond H'$ when $d(H) = d(H') = n$ and there exists a representative S of H and a representative S' of H' such that for all $\sigma \in \mathfrak{S}_n$ we have $(\sigma \cdot S) \diamond (\sigma \cdot S')$. In addition, if the relation \diamond satisfies

$$\forall S, S' \in \mathcal{P}_h(\mathbb{A}) \quad S \diamond S' \Rightarrow d(S) = d(S')$$

then the quotient map is compatible with \diamond and its extension. The relation of inclusion \subseteq over $\mathcal{P}_h(\mathbb{A})$ obviously satisfies these properties and therefore extends to $\mathcal{H}(\mathbb{A})$.

3.5 Cubical Areas

A **cube** of dimension n is a word of length n on the alphabet \mathcal{I} of *non-empty* intervals of \mathbb{R} . The elements of $\mathcal{H}(\mathcal{I})$ are called the **cubical coverings**. Furthermore the homogeneous monoid $\mathcal{H}(\mathcal{I})$ is endowed with a preorder arising from the inclusion on \mathcal{I} . Indeed, given two sets of cubes of the same length S and S' we write $S \preceq S'$ when for all cubes $C \in S$ there exists a cube $C' \in S'$ such that $C \subseteq C'$. The relation \preceq provides the monoid $\mathcal{P}(\mathcal{I})$ with a preorder that can be extended to $\mathcal{H}(\mathcal{I})$ by setting $H \preceq H'$ when $d(H) = d(H') = n$ and there exists a representative S of H and a representative S' of H' such that for all $\sigma \in \mathfrak{S}_n$ we have $(\sigma \cdot S) \preceq (\sigma \cdot S')$. We now establish a Galois connection between $(\mathcal{H}(\mathbb{R}), \subseteq)$ and $(\mathcal{H}(\mathcal{I}), \preceq)$. Given a cubical covering F we define $\gamma(F)$ as

$$\left\{ \bigcup_{C \in S} C \mid S \in F \right\}$$

Furthermore γ is a morphism of monoids and if $F \preceq F'$ then $\gamma(F) \subseteq \gamma(F')$. Conversely, given some S in $\mathcal{P}_h(\mathbb{R}^*)$ the collection of n -dimensional cubes C such that $C \subseteq S$, ordered by inclusion, is a semilattice whose maximal elements are called the **maximal cubes** of S . The set M_S of maximal cubes of S is homogeneous and for all $\sigma \in \mathfrak{S}_n$, $\sigma \cdot M_S = M_{\sigma \cdot S}$. Then given $H \in \mathcal{H}(\mathbb{R})$ we define $\alpha(H)$ as

$$\left\{ M_S \mid S \in H \right\}$$

Furthermore α is a morphism of monoids and if $H \subseteq H'$ then $\alpha(H) \subseteq \alpha(H')$. Then we have a Galois connection:

Proposition 1. $\gamma \circ \alpha = \text{id}_{\mathcal{H}(\mathbb{R})}$ and $\text{id}_{\mathcal{H}(\mathcal{I})} \preceq \alpha \circ \gamma$.

Given $H \in \mathcal{H}(\mathbb{R})$ and $F \in \mathcal{H}(\mathcal{I})$ we say that F is a cubical covering of H when $\gamma(F) = H$. The **cubical areas** are the elements H of $\mathcal{H}(\mathbb{R})$ which admit a *finite* cubical covering. The collection of cubical areas (resp. finite cubical coverings) forms a sub-monoid **Are** of $\mathcal{H}(\mathbb{R})$ (resp. **Cov** of $\mathcal{H}(\mathcal{I})$). The restrictions of the morphisms γ and α to **Cov** and **Are** induce another Galois connection.

Proposition 2. $\gamma \circ \alpha = \text{id}_{\text{Are}}$ and $\text{id}_{\text{Cov}} \preceq \alpha \circ \gamma$.

Moreover, the morphisms γ and α of Proposition 2 induce a pair of isomorphisms of commutative monoids between **Are** and the collection of fixpoints of $\alpha \circ \gamma$. A submonoid of a free commutative monoid may not be free. Yet, under a simple additional hypothesis this pathological behaviour is no more possible. We say that a submonoid P of a monoid M is **pure** when for all $x, y \in M$, $x * y \in P \Rightarrow x \in P$ and $y \in P$.

Lemma 1. *Every pure submonoid of a free commutative monoid is free.*

Proof. Let P be a pure submonoid of a free commutative monoid M . Let p be an element of P written as a product $x_1 \cdots x_n$ of irreducible elements of M . Each

x_i is obviously an irreducible element of P so any element of P can be written as a product of irreducible elements of P . Furthermore any irreducible element of P is also an irreducible element of M because P is pure in M . It follows that any elements of P can be written as a product of irreducible elements of P in a unique way i.e. P is free.

Then we have:

Theorem 2. *The commutative monoid of cubical areas is free and has infinitely many irreducible elements.*

Proof. Let X and X' be two elements of $\mathcal{H}(\mathbb{R})$ and suppose $X * X'$ belongs to \mathbf{Are} . Since both α and γ are morphisms of monoids we have $\alpha \circ \gamma(X * X') = \alpha \circ \gamma(X) * \alpha \circ \gamma(X')$ which is finite. It follows that both $\alpha \circ \gamma(X)$ and $\alpha \circ \gamma(X')$ are finite. Hence X and X' actually belongs to \mathbf{Are} , which is thus free as a pure submonoid of $\mathcal{H}(\mathbb{R})$.

4 Effective Factoring of Cubical Areas

Beyond their theoretical usefulness, the maximal cubes provide the data structure which allows to handle algorithmically cubical areas, as in the static analyzer **ALCOOL** which is devoted to the study of parallel programs.

4.1 Implementation

We need an algorithm which performs decompositions in $\mathcal{H}(\mathbb{A})$, its implementation is directly based on the proof of the Theorem 1: $H \in \mathcal{H}(\mathbb{A})$ is reducible if and only if there exists some representative S of H which admits a non trivial decomposition in $\mathcal{P}_h(\mathbb{A}^*)$. In order to describe the algorithm we define

$$S \circ A := \{w \circ A \mid w \in S\}$$

for any $S \in \mathcal{P}_h(\mathbb{A}^*)$ and $A \subseteq \{1, \dots, d(S)\}$. Moreover for $w' \in \mathbb{A}^*$ with $\ell(w') \leq d(S)$, and A^c the complement of A (in $\{1, \dots, d(S)\}$), we define the set of words

$$\Psi(w', A, S) := \{w \circ A^c \mid w \in S \text{ and } S \circ A = w'\}$$

Then the class $[S \circ A] \in \mathcal{H}(\mathbb{A})$ divides H if and only if for all $w' \in S \circ A$ one has $\Psi(w', A, S) = [S \circ A^c]$. In the monoid $\mathcal{H}(\mathbb{A})$ we thus have

$$[S \circ A] * [S \circ A^c] = H$$

Then we look for some divisor of H by testing all the non empty subsets A of $\{1, \dots, d(S)\}$ according to the following total ordering

$$A \leq A' \text{ when } |A| < |A'| \text{ or } (|A| = |A'| \text{ and } |A| \sqsubseteq_{\text{lex}} |A'|)$$

where \sqsubseteq_{lex} is the lexicographic ordering. Doing so, we know that if A is the first value such that $[S \circ A]$ divides H , then $[S \circ A]$ is irreducible. Moreover we have $d([S \circ A]) = |A|$ and for all $H_0, H_1 \in \mathcal{H}(\mathbb{A})$, $d(H_0 * H_1) = d(H_0) + d(H_1)$ hence we can suppose

$$|A| \leq \frac{d(H)}{2} + 1$$

The software **ALC00L** is entirely written in **OCaml**. The complexity of the decomposition algorithm implemented in it is exponential in the dimension n of the cubical area since it checks all the subsets of $\{0, \dots, n-1\}$. However the computation time actually devoted to the decomposition is rather small with regard to the global execution time required by the whole analysis. Indeed the algorithm which builds the state space of the program, though it has the same theoretical complexity as the decomposition algorithm, has to handle heavier structures.

4.2 A detailed example

We treat the case of the program Σ given in Example 1. Its geometric model is given on the left hand side of Figure 3. Applying the permutation $(2, 3)$ we obtain the right hand side set.

Fig. 3. Cubical area of Example 1

$[0, 1[*[0, 1[*[0, -[*[0, -[$	$[0, 1[*[0, -[*[0, 1[*[0, -[$
$ [0, 1[*[4, -[*[0, -[*[0, -[$	$ [0, 1[*[0, -[*[4, -[*[0, -[$
$ [0, 1[*[0, -[*[0, -[*[0, 1[$	$ [0, 1[*[0, -[*[0, -[*[0, 1[$
$ [0, 1[*[0, -[*[0, -[*[4, -[$	$ [0, 1[*[0, -[*[0, -[*[4, -[$
$ [4, -[*[0, 1[*[0, -[*[0, -[$	$ [4, -[*[0, -[*[0, 1[*[0, -[$
$ [4, -[*[4, -[*[0, -[*[0, -[$	$ [4, -[*[0, -[*[4, -[*[0, -[$
$ [4, -[*[0, -[*[0, -[*[0, 1[$	$ [4, -[*[0, -[*[0, -[*[0, 1[$
$ [4, -[*[0, -[*[0, -[*[4, -[$	$ [4, -[*[0, -[*[0, -[*[4, -[$
$ [0, -[*[0, 1[*[0, 1[*[0, -[$	$ [0, -[*[0, 1[*[0, 1[*[0, -[$
$ [0, -[*[0, 1[*[4, -[*[0, -[$	$ [0, -[*[4, -[*[0, 1[*[0, -[$
$ [0, -[*[0, -[*[0, 1[*[0, 1[$	$ [0, -[*[0, 1[*[0, -[*[0, 1[$
$ [0, -[*[0, -[*[0, 1[*[4, -[$	$ [0, -[*[0, 1[*[0, -[*[4, -[$
$ [0, -[*[4, -[*[0, 1[*[0, -[$	$ [0, -[*[0, 1[*[4, -[*[0, -[$
$ [0, -[*[4, -[*[4, -[*[0, -[$	$ [0, -[*[4, -[*[4, -[*[0, -[$
$ [0, -[*[0, -[*[4, -[*[0, 1[$	$ [0, -[*[4, -[*[0, -[*[0, 1[$
$ [0, -[*[0, -[*[4, -[*[4, -[$	$ [0, -[*[4, -[*[0, -[*[4, -[$

Then we can check that the (right hand side of Figure 3) cubical area can be written as

$$\left([0, 1[*[0, -[\parallel [4, -[*[0, -[\parallel [0, -[*[0, 1[\parallel [0, -[*[4, -[\right)^2$$

Then we have

$$(2, 3) \cdot \{\{1, 2\}, \{3, 4\}\} = \{\{1, 3\}, \{2, 4\}\}$$

and it follows that in the program Σ the sets of processes $\{\pi_1, \pi_3\}$ and $\{\pi_2, \pi_4\}$ run independently from each other.

4.3 Benchmarks

We describe some programs upon which the algorithm has been tested. The program Σ_{n_1, \dots, n_k} is made of k groups of processes: for all $i \in \{1, \dots, k\}$ it contains n_i copies of the process

$$P(a_i).P(b).V(b).V(a_i)$$

where a_i is a mutex and b is a semaphore of arity $k + 1$. All processes then share the resource b , but as for Σ in Example 1 the k groups are actually independent. On the other hand the program $\Sigma'_{n_1, \dots, n_k}$ is the same as Σ_{n_1, \dots, n_k} but with b of arity only k , which forbids any decomposition. The n -philosophers programs implement the standard n dining philosophers algorithm.

The benchmark table of Figure 4 has been obtained using the Unix command `time` which is not accurate. Hence these results have to be understood as an over-approximation of the mean execution time. It is also worth remarking that our

Fig. 4. Benchmarks

Example	Time (in sec.)	Decomp.
6 philosophers	0.2	No
7 philosophers	0.7	No
8 philosophers	3.5	No
9 philosophers	21	No
10 philosophers	152	No

Example	Time (in sec.)	Decomp.	Example	Time (in sec.)	Decomp.
$\Sigma_{2,2}$	0.1	$\{1, 3\}\{2, 4\}$	$\Sigma'_{2,2}$	0.1	No
$\Sigma_{2,2,2}$	0.1	$\{1, 4\}\{2, 5\}\{3, 6\}$	$\Sigma'_{2,2,2}$	0.3	No
$\Sigma_{3,3}$	0.13	$\{1, 3, 5\}\{2, 4, 6\}$	$\Sigma'_{3,3}$	0.52	No
$\Sigma_{2,2,2,2}$	0.13	$\{1, 5\}\{2, 6\}\{3, 7\}\{4, 8\}$	$\Sigma'_{2,2,2,2}$	7.1	No
$\Sigma_{4,4}$	1	$\{1, 3, 5, 7\}\{2, 4, 6, 8\}$	$\Sigma'_{4,4}$	33	No
$\Sigma_{3,3,3}$	1.5	$\{1, 4, 7\}\{2, 5, 8\}\{3, 6, 9\}$	$\Sigma'_{3,3,3}$	293	No
$\Sigma_{4,5}$	6.1	$\{1, 3, 5, 7\}\{2, 4, 6, 8\}$	$\Sigma'_{4,5}$	327	No
$\Sigma_{5,5}$	50	$\{1, 3, 5, 7, 9\}\{2, 4, 6, 8, 10\}$	$\Sigma'_{5,5}$	2875	No

algorithm is efficient when the cubical area to decompose is actually a cartesian product of several irreducible cubical areas of small dimension. This remark should be compared with the fact that the standard decomposition algorithm of integer into primes is very efficient on products of small prime numbers.

5 Conclusion

Related work.

The problem of decomposition of concurrent programs in *CCS*-style has been studied in [GM92] and [MM93]. By the possibility of using semaphores of arbitrary arity, our work seems to go beyond this previous approach. Also note that the silent and synchronous communication mechanism of *CCS* can be given a straightforward geometric interpretation which falls in the scope of the present discussion. However, the link between bisimilarity in *CCS* and isomorphic geometric interpretations is still to be explored to make clear the relations between these works.

In [LvO05] *B. Luttik* and *V. van Oostrom* have characterized the commutative monoids with unique decomposition property as those which can be provided with a so-called decomposition order. In the case where the property holds, the divisibility order always fits. Yet, there might exist a more convenient one. Unfortunately, in the current setting the authors are not aware of any such order yielding direct proofs. Nevertheless it is worth noticing that this approach is actually applied for decomposition of processes in a normed ACP theory for which a convenient decomposition order exists.

Conclusion.

This paper uses a geometric semantics for concurrent programs, and presents a proof of a unique decomposition property together with an algorithm working at this semantic level (Theorem 2). The main strength of this work is that it applies to any concurrent program yielding a cubical area. Example of features allowed in this setting are: semaphores, synchronisation barriers, synchronous as well as asynchronous communications (with finite or infinite message queues), conditional branchings. In fact we can even consider loops provided we replace the set \mathcal{I} of intervals of the real line \mathbb{R} by the set \mathcal{A} of arcs of the circle.

Future work.

Actually, a cubical area naturally enjoys a pospace⁹ structure. Pospaces are among simplest objects studied in *Directed Algebraic Topology*. In particular, a cubical area is associated with its category of components [FGHR04,GH05,Hau06] and [GH07], which is proven to be finite, loop-free¹⁰ and in most cases connected. Then, as the cubical areas do, these categories together with cartesian product form a free commutative monoid. It is worth noticing this is actually the generalization of a result concerning finite posets which has been established in the early fifties [Has51]. Therefore a program Π can be decomposed by lifting the decomposition of the category of components of its geometric model $\llbracket \Pi \rrbracket$. In general, the relation between the decomposition of a cubical area and the one of its category of components is a theoretical issue the authors wish to investigate.

Another important concern is a clarification of the control constructs compatible with cubical areas: replacing in some dimensions the intervals of the real

⁹ shorthand for “partially ordered spaces” [Nac65].

¹⁰ Loop-free categories were introduced in [Hae91,Hae92] as “small categories without loop” or “scwols”.

line by the arcs of the circle as mentioned above corresponds to a global loop, but some richer structures may be useful.

A final point of interest is the investigation of the exact relation between our semantic results and the syntactic ones of [GM92,MM93,LvO05]. Indeed they use *CCS*-like syntaxes to describe some classes of edge-labelled graphs modulo bisimilarity, whereas the category of components of our models correspond to some other graphs modulo directed homotopy. Hence the question: what is in this setting the relation between bisimilarity and homotopy?

References

- [BMP99] O. Bournez, O. Maler and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *Hybrid Systems: Computation and Control*. Springer, 1999.
- [CR87] S. D. Carson and P. F. Reynolds Jr. The geometry of semaphore programs. *ACM Transactions on Programming Languages and Systems*, 9(1):25–53, 1987.
- [Dij68] Edsger W. Dijkstra. Cooperating sequential processes. In *Programming Languages: NATO Advanced Study Institute*, pages 43–112. Academic Press, 1968.
- [FGHR04] Lisbeth Fajstrup, Eric Goubault, Emmanuel Haucourt and Martin Raoulen. Component categories and the fundamental category. *APCS*, 12(1):81–108, 2004.
- [GH05] E. Goubault and E. Haucourt. A practical application of geometric semantics to static analysis of concurrent programs. *CONCUR'05, LNCS* 3653, 2005.
- [GH07] Eric Goubault and Emmanuel Haucourt. Component categories and the fundamental category II. *APCS*, 15(4), 2007.
- [GM92] Jan Friso Groote and Faron Moller. Verification of Parallel Systems via Decomposition. *CONCUR '92*, 62–76, 1992.
- [Hae91] André Haefliger. Complexes of groups and orbihedra. In *Group theory from a geometrical viewpoint*, pp. 504–540. World Scientific, 1991.
- [Hae92] André Haefliger. Extension of complexes of groups. *Annales de l'institut Fourier*, 42(1-2):275–311, 1992. <http://www.numdam.org/>
- [Has51] Junji Hashimoto. On direct product decomposition of partially ordered sets. *Annals of Mathematics*, (54):315–318, 1951.
- [HN50] Junji Hashimoto and Tadasi Nakayama. On a problem of Garrett Birkhoff. In *Proceedings of the American Mathematical Society*, volume 1, pp. 141–142, 1950.
- [Hau06] Emmanuel Haucourt. Categories of components and Loop-free categories. *Theory and Applications of Categories*, 16(27):736–770, 2006.
- [Lan02] Serge Lang Algebra, 4th corrected printing. *Graduate Texts in Mathematics*. Springer, 2002.
- [LvO05] B. Luttik and V. van Oostrom. Decomposition orders: another generalisation of the fundamental theorem of arithmetic. *TCS*, 335(2-3):147–186, 2005.
- [MM93] Robin Milner and Faron Moller. Unique Decomposition of Processes. *TCS*, 107(2):357–363, 1993.
- [Nac65] Leopoldo Nachbin. *Topology and Order*, volume 4 of *Van Nostrand Mathematical Studies*. Van Nostrand, Princeton, 1965.
- [Tha09] Dang Thao. Methods and Tools for Computer Aided Design of Embedded Systems. HDR Thesis, Chapter 5. 2009.
- [Win95] Glynn Winskel. Handbook of Logic in Computer Science vol.4 : Semantic Modelling. Chapter 1. Oxford University Press, 1995.